Reconhecimento Sintático

Luiz Paulo Grafetti Terres (2121101010)[1]

[1]Universidade Federal da Fronteira Sul

Julho 2024

1 Resumo

Este trabalho apresenta detalhes teóricos da etapa de compilação responsável pelo reconhecimento sintático, e discute a implementação de um *parser* sintático, tarefa requisitada no componente curricular de Construção de Compiladores, ministrada pelo Professor Dr.Braulio Adriano de Mello. A implementação discutida é do autor, e está disponível em: `https://github.com/luizpgt/Compiladores_2024.1.git`.

2 Introdução

Em Construção de Compiladores, é um elemento comum para algumas etapas de compilação, um "arquivo de programa", que possui palavras reservadas, identificadores e outros símbolos especiais. Neste trabalho, trataremos por arquivo de programa, um arquivo qualquer com entradas que busquem estar em conformidade com uma **linguagem regular** definida.

A etapa de Reconhecimento Léxico, discutida e implementada na última entrega da disciplina, é responsável pela identificação e classificação do conteúdo presente no arquivo de programa. Essa etapa verifica se existem erros lexicais presentes no arquivo, o que impossibilitaria as próximas etapas de compilação (como reconhecimento sintático) de produzir os resultados esperados.

O Reconhecedor Sintático deste trabalho busca fazer *parsing* com base em Gramáticas Livres de Contexto genéricas (ou, à escolha do usuárioo), necessitando-se assim ter acesso a uma Tabela LALR, que definirá as ações/transições e saltos necessários para o reconhecimento sintático.

2.1 Problema

Necessidade de **Reconhecimento Sintático** a partir de fitas de saída de etapas de compilação anteriores, e disposição dos resultados encontrados para as próximas etapas de compilação.

2.2 Objetivo

Implementar um Reconhecedor Sintático, capaz de - a partir de uma fita de saída obtida do projeto anterior da disciplina - classificar o conteúdo de um arquivo de programa genérico (que esteja de acordo com as regras Léxicas e Sintáticas da GR (último projeto, análise léxica) e GLC (este projeto, análise sintática)) como sintáticamente correto ou incorreto. Caso o programa do usuário esteja inconsistente, o trabalho apresentará os erros necessários para que seja possível identificar o erro sintático no arquivo de programa de entrada.

3 Referencial Teórico

Este trabalho trata da etapa de compilação da análise sintática. Essa etapa verifica a concordância de uma fita de saída (resultado da etapa de reconhecimento léxico) com as regras de uma Gramática Livre de Contexto. Um analisador sintático, como o implementado, é um programa responsável por ler a fita de saída contendo informações de identificação, rótulo original da palavra reconhecida pela etapa léxica, e número de linha onde se encontra, e fazer a verificação da ordem dos componentes que formam a fita.

A etapa de análise sintática depende diretamente do resultado obtido durante a análise léxica. A solução desse projeto utiliza o último trabalho implementado nesta disciplina, que faz o reconhecimento léxico, e disponibiliza para as etapas posteriores as informações necessárias na fita de saída / tabela de símbolos.

É necessário para o reconhecimento da fita de saída, a passagem dos símbolos por uma Tabela de Parsing. Nesse trabalho são utilizadas tabelas LALR, resultantes do programa GoldParser `http://www.goldparser.org/`. Uma tabela consiste basicamente em estados da tabela (linhas), e símbolos da tabela (símbolos terminais e não terminais da GLC). Símbolos não terminais, em certos estados, apresentam **saltos**, e símbolos terminais apresentam ações e transições na tabela, também de acordo com seu estado. As ações/transições são compreendidas por **empilhamentos** e **reduções**, que aumentam e diminuem a quantidade de itens na pilha, respectivamente. Reduções não alteram a quantidade de itens lidos da fita. Saltos, por outro lado, também não alteram a quantia de elementos reconhecidos na fita, mas determinam qual será o próximo estado da tabela que guiará a próxima ação.

O resultado da etapa de análise sintática produz informações que são diretamente relevantes para a próxima etapa de compilação, que envolve análise semântica. Na análise semântica, o arquivo de programa é avaliado em relação ao significado dos elementos presentes nele, buscando por exemplo, otimizações.

4 Implementação e Resultados

Para melhor entendimento dessa seção, recomenda-se que o leitor tenha acesso à implementação do autor. Com ela, poderá testá-la e perceber mais detalhes de implementação.

4.1 Definição da Linguagem

As palavras que serão aceitas no arquivo de programa, precisam encontrar-se no arquivo `\syntactic\_analysis\finite_automata_input.txt"`. Neste arquivo, deverão ser definidos os tokens da linguagem, e suas gramáticas regulares. Com base no trabalho desenvolvido na disciplina de Linguagens Formais e Autômatos, as entradas nesse "arquivo da linguagem" serão tratadas, e serão definidas em um único Autômato Finito Determinístico.

Esta parte do projeto está coberta no Submódulo `Deterministic_finite_automaton`, discutido na disciplina de Linguagens Formais e Compiladores, e não será detalhado nesse trabalho. Exemplo de valores no arquivo da linguagem:

```
¨ This is a comment !
¨ This files contains regular grammars and tokens.
¨ It is meant to guaranteed work with at most:
¨ 1 Regular grammar, with N rules
¨ N Tokens of any reasonable length

plus
times
popen
pclose

<S> ::= 0<A> | 1<A> | 2<A> | 3<A> | 4<A> | 5<A> | 6<A> | 7<A> | 8<A> | 9<A>
<A> ::= 0<A> | 1<A> | 2<A> | 3<A> | 4<A> | 5<A> | 6<A> | 7<A> | 8<A> | 9<A> | E
```

4.2 Análise léxica

A etapa do analisador léxico, tem como entrada o arquivo de programa, `\example_program_file.txt"`. Nesse arquivo estarão presentes as palavras que serão classificadas, e ao final da etapa de análise léxica, estará produzida uma Fita de Saída, e uma Tabela de Símbolos.

É importante notar que a implementação da fase de análise léxica se baseia em um modelo. Esse modelo é uma classe chamada `Lexical_Analyzer`, e possui como atributos mais importantes a tabela determinística de transições (do autômato finito determinístico), uma **fita de saída** e uma **tabela de símbolos**. No modelo também são definidas questões úteis menores, como o **símbolo final de sentença**, e os **estados inicial** e de **erro**.

A implementação da análise léxica não será discutida nesse trabalho. O arquivo de programa, entrada da etapa sintática, é da forma:

```
popen 123 pclose
plus 455
times 2
plus 529
times 1
```

O resultado da chamada dessa etapa de compilação, gera um objeto do modelo `Lexical_Analyzer`, com uma tabela de símbolos e fita de saída. Esses resultados são dispostos pelo programa no arquivo `\program_output_table.txt"` :

```
¨ popen
¨ 1_var_A
¨ pclose
¨ plus
¨ times
¨ $ (EOF)
popen 1 popen
1_var_A 1 123
pclose 1 pclose
plus 1 plus
1_var_A 1 455
times 1 times
1_var_A 1 2
plus 1 plus
1_var_A 1 529
times 1 times
1_var_A 1 1
$
```

"Traduções" para alinhar detalhes da GR e GLC podem ser alcançadas modificando esse arquivo, nas linhas iniciadas por "¨". Por exemplo, uma gramática de operadores pode esperar um identificador **id** ao invés de **1_var_A**, nesse caso, a segunda linha torna-se **¨ 1_var_A id**.

4.3 Análise sintática

Como mencionado anteriormente nesse projeto, a etapa da análise sintática consiste basicamente na verificação da ordem dos identificadores dispostos no programa fonte. Para essa verificação, é usado nesse trabalho uma solução utilizando uma **Tabela LALR**. Essa tabela foi obtida por meio do software GoldParser `http://www.goldparser.org/`. Ao ser alimentado com uma GLC à escolha do usuário, o resultado obtido é uma tabela LALR válida.

O principal algoritmo implementado neste trabalho trata-se do reconhecimento sintático por meio da tabela LALR, fazendo as ações/transições e saltos necessários para validar ou não a fita de saída.

Data: file: Tabela de símbolos da etapa de compilação anterior.
Result: Sentença sintaticamente correta (True) OU mensagem de erro (False).

```
TS ← Tabela de símbolos
RULES ← Produções de GLC numeradas
STACK ← [0] ;      // Pilha inicialmente com o estado inicial
LALR_TABLE ← Tabela LALR
for tableel in LALR_TABLE do
    stackel ← topo do STACK
    encontra_acao_na_tabela(stackel, tableel)
    if existe ação na posição da tabela then
        if encontrado estado de aceite then
            Retorna True
        end
        else
            Performa Empilhamento OU Redução
        end
    end
    else
        Exibe mensagem de erro
    end
end
Retorna False
```

Algorithm 1: Parsing da Fita/Tabela de Símbolos na Tabela LALR

5 Conclusões

Em síntese do que foi discorrido neste trabalho, pode-se notar que o objetivo deste trabalho foi alcançado. A implementação prática de um Reconhecedor Sintático sobretudo agrega ao aluno um entendimento mais completo da teoria apresentada em sala de aula. Há espaço para melhorias na implementação do autor, que tem objetivos didáticos e deixa a desejar em alguns aspectos como otimização do tempo de execução. Outra melhoria seria ajustar o programa de análise léxica de forma que fosse possível alinhar a GLC à GR sem ser necessário ajustes manuais intermediários.